|QSESC>

Quantum Software Engineering Standardization Council

UNIVERSALIZATION OF QUANTUM SOFTWARE ENGINEERING

SheQuantum's Quantum Software Engineering Standardization Council (QSESC)

Authors and Co-Contributors:

Nithyasri Srivathsan, SheQuantum, *Singapore*

Elías Fernández-Combarro Álvarez, University of Oviedo, CERN, *Spain*

Zeynep Koruturk, Goldman Sachs, *United Kingdom*

Eduardo Henrique Matos Maschio (Pasqal) *Netherlands*

Jennifer E. Decker, National Research Council Canada, *Canada*

Aida Todri-Sanial, Eindhoven University of Technology, *Netherland,* CNRS, *France*

Contents

Universalization of Quantum Software Engineering

The recent accelerated interest in quantum computing and engineering efforts towards the development of quantum hardware devices has motivated the need for devising algorithms to tap into the potential that quantum computing can offer for real world applications through software. Quantum computing exploits the non-intuitive laws of quantum mechanics like superposition, interference, and entanglement to perform computation using quantum bits. This offers parallelism and computational speedups enabling it to tackle some limitations of classical computing in solving certain difficult problems in domains like finance, pharmaceutics and bio-medicine, molecular chemistry, material sciences, and optimization of logistics and supply chain. This very special nature of quantum computation also makes the designing and engineering of quantum software, a non-trivial task. In order to convert our theoretical knowledge of the emerging field of quantum computing into useful real-world applications, the ability to build algorithms and implement them as software programs becomes crucial. However, the current state of quantum software lacks the universal acceptance of standard guidelines, methods, and principles, leading to inefficient and disjoint efforts in developing quantum software. As history has it, concepts in classical software engineering have evolved from the 1920s and this era demands the need for this history to repeat for the promising art of quantum software engineering!

This whitepaper dives deep into Quantum Software Engineering, discussing the need for universalization and the potential pathway towards a standard for QSWE by analyzing the challenges, stakeholders, technological capabilities, and proposes a simple Quantum Software Development Lifecycle (QSDLC) model, calling for universalization of quantum software engineering and advocates for the development of a standardized hardware-agnostic and programming-language-agnostic approach to writing scalable, maintainable, interoperable, reliable, and efficient quantum software programs. In this whitepaper, we define the term "universalization" as finding a common ground among the contributions being made to the field in terms of being hardware and programming language agnostic, to guide efforts towards the ultimate goal of making quantum computers useful. We understand and in fact, encourage that this definition must evolve to a more specific and narrower context as the field progresses.

Objectives of this Whitepaper:

In this whitepaper, we aim to explore the following questions:

- What is Quantum Software Engineering (QSWE)?
- Why do we need a universal standard for QSWE?
- Can we take inspiration from Classical Software Engineering? How different are Classical and Quantum SWE?

- What can be a good model for QSDLC in a bigger picture perspective?
- Call for action: How can we approach the Universalization of QSWE? Who are the stakeholders?

What is QSWE?

Quantum Software Engineering or QSWE is broadly the science of engineering a structured approach to design, implement, maintain, and evolve software for quantum computers based on systematic guidelines, principles, and theories.

Implementation of quantum software as quantum algorithms is needed in order to make quantum computing practical and useful for real-life use cases. Implementation of quantum algorithms must be inspired from classical software engineering techniques to ensure it's reliable. Moreover, it's productive to learn from and adopt the concept of a software development life cycle (SDLC) for writing quantum software, involving the primary phases that we're familiar with from classical SDLC – requirements analysis, architecture and design, development, testing, debugging, and maintenance. Such a SDLC for quantum software doesn't currently exist and can be one of the stepping stones towards developing homogeneity in our approach to QSWE. [1]

Ideally, Quantum Software Engineering is the art and science of being able to write software which is purely quantum in nature, to be executed on a purely quantum device. It's obvious that this may not only be an overly-ambitious endeavour but scientists haven't been able to prove that this might be a possibility. Thus, in this whitepaper, we will not consider this ideal approach to quantum software engineering.

A more practical approach, currently accepted by the quantum industry is the Classical-Quantum Hybrid approach which involves fostering an interactive collaboration that sends specific parts of the problem to the best suited solver. This leverages the advantage that classical computing offers at specific stages and that quantum computing offers at other stages, to ultimately solve the problem iteratively. This has been found to be a good way to reach the potential of a near-term quantum computer as beautifully conveyed in various works including that by (Callison & Chancellor, 2022). The need for the coexistence of classical and quantum computing to enable near-term utility is unequivocal.

Another interesting practical approach to quantum computation is a technique called quantum simulation. It would not be wrong to say that simulation is a prototype to gain insights into how a real quantum computation might work using techniques like tensor networks or Monte Carlo methods. This might also be a temporary solution to the lack of quantum hardware resources. Quantum simulations, however, is quintessential in the sense that it does not contain noise intrinsically. However, to navigate this, it is possible to build noise models which can then be artificially introduced into the system for practical purposes, to be able to simulate closest to how natural systems behave.

Quantum simulations coupled with High Performance Computing (HPC) is essential for developing hybrid quantum-classical algorithms, such as variational quantum eigensolvers (VQEs) and quantum approximate optimization algorithms (QAOAs) which form the fundamental base of many industrial applications in pharma, biomedical, finance, logistics, chemistry, to mention few examples.

In both cases, it can be clearly identified that the key challenge in the domain of quantum software engineering is to be able to rework and provide an extension to the whole of classical software engineering for a quantum mechanical domain so that programmers can manipulate quantum programs with the same ease and confidence that they manipulate today's classical programs. [2]

Quantum computing is not only to be perceived as a technological advancement, but also needs to be looked upon as an enterprising opportunity for developing and standardizing a new general-purpose paradigm of computation and more importantly, a strikingly unique realm of software development. This calls for new, quantum-specific, software engineering approaches. [3] Quantum software engineering methods provide the techniques for constructing quantum software. They consist of a wide range of tasks, including the design of data structures, program architecture, algorithm procedure, coding, testing, and maintenance. Quantum software engineering tools also provide automated or semiautomated support for these methods.

QSWE processes are the foundation for quantum software engineering. Processes provide the glue that holds the methods and tools together and enables the rational and timely development of quantum software. They define the sequence in which methods would be applied: the deliverables, the controls that help quality assurance and change coordination, as well as the milestones that enable quantum software managers to access the progress. [3]

In a special issue of Applied Sciences (ISSN 2076-3417) titled 'Quantum Software Engineering and Programming', by Professor Mario Piattini highlights the current state of quantum software development. "With the rise of the first quantum computers, several programming languages and quantum algorithms came up with promising results. Nevertheless, quantum software is not yet produced in a rigorous and industrial way. Software engineering and programming practices need to be brought into the domain of quantum computing" (Piattini, 2023). [4]

It's clear that origin of any discussion on quantum software must be with the quantum stack which might range from programming languages to compilers to operating systems in a top-down approach.

This software stack requires tasks such as the design of quantum programs, implementation techniques for quantum algorithms, and testing and maintenance of quantum software at the topmost. But the inherent quantum characteristics of superposition and entanglement, disenable the usage of classical software engineering methods for practical quantum software. In addition, building quantum programs

poses significant difficulties for software developers due to the need for switching to an entirely different programming mindset with such counterintuitive quantum principles. (Chen & Schoute, 2022)

Thus, it becomes important that our entry-point to the quantum software stack, must be programming-language agnostic, as a first step towards universalization. This would mean that a programmer with experience of Python or Java or any other language should be able to utilize their pre-existing skills to build quantum programs without the necessity to learn a particular language they might not be familiar with already. This ensures that the primary barrier to contributing to quantum computing is broken down.

However, this alone does not mean that the developer would be free from the complexities and challenges posed by the quantum mechanical laws governing quantum computation. A potential solution to this problem would be to adopt, from the world of classical software engineering, the principle of abstraction to hide out the unnecessary details and deeper concepts of quantum physics which might be used in quantum programs.

[5] Quantum computing being a young field, it is quite natural that very minimal work has been done to determine the challenges quantum software developers face and will face in the future as technology evolves. This step will play an important role in formulation of industry best practices. “To the best of knowledge, no empirical study has been conducted to explore the key challenging factors faced by QSE practitioners. Hence, we are motivated to conduct a study that can help the software organization to focus on challenging factors that could have significant negative influence on successful execution of QSE process.” (Liu & Yang, 2023)

Although it is not clear what are all the challenges and opportunities of quantum computing facing the software engineering community, we have identified that lack of a programming-language-agnostic approach and universally accepted levels of abstraction could be key challenges that quantum computing researchers and engineers presently face.

In this section, we have surveyed the difficulties of quantum software engineering and hinted towards potential solutions, which we will discuss in more detail in the coming sections. We argue that there is an urgent need for standardization that is independent of hardware and programming languages, that standardization organizations, in collaboration with researchers from academia, industry, and quantum hardware providers, must work to develop more rigorous and efficient ways of expressing useful quantum programs for applications that will serve humankind. As discussed, since the majority of classical software engineering standards do not apply to QSWE, it is essential to include and involve all relevant stakeholders in the quantum community in order to create a standard for QWSE and pave the path towards universalization. This will ensure that the developers of quantum software only need to adhere to that process rather than worry about various arrays of quantum technologies, hardware, or toolkits, that one can choose from at the moment.

We, therefore, propose that there exists a dire need for a sustained effort towards creating a standard to formulate a layer interfacing the quantum software architecture (QSA) and quantum hardware architecture (QHA). The Quantum Software Engineering Life Cycle Model (shown in Fig 1) is thus proposed towards such a QSA-QHA standard.

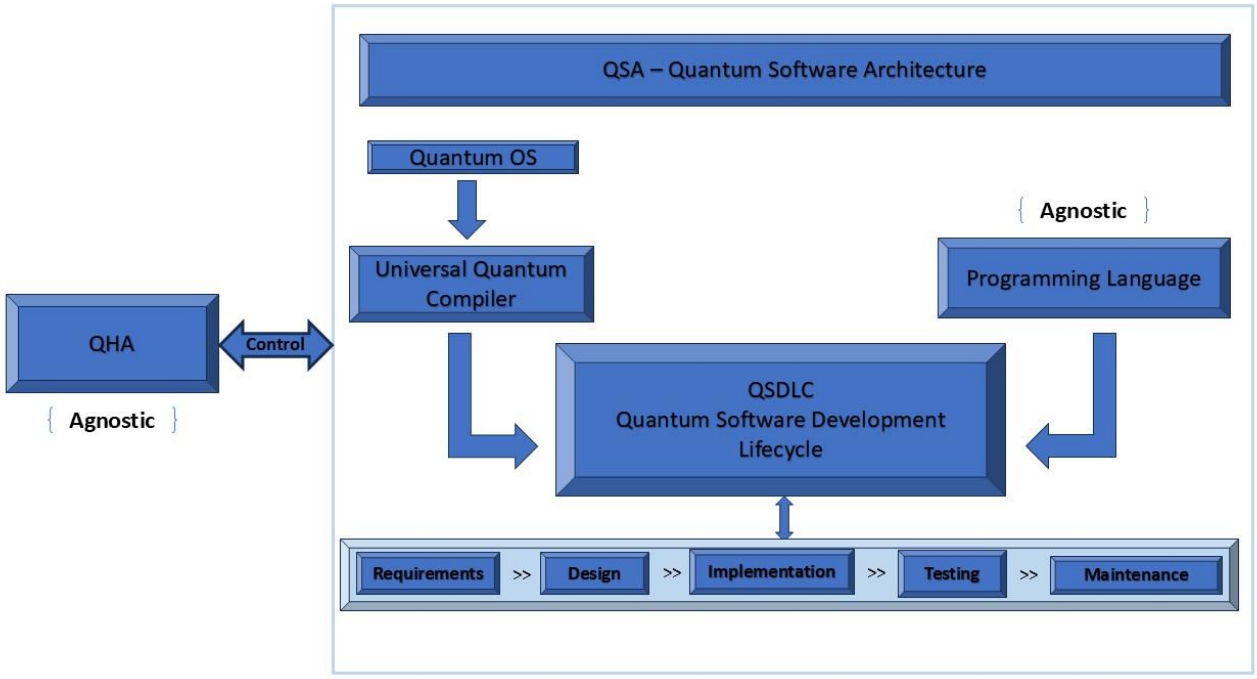


Fig. 1 QSDLC, QHA, and QSA: The Bigger Picture

Diagram Terminology:

Quantum Hardware Architecture (QHA):

QHA is the blueprint of how a quantum device is organized such that its components work well together to perform tasks like quantum computation, quantum memory management, and input and output (I/O) management.

Quantum Software Architecture (QSA):

QSA is a more abstract concept that dictates how decisions are made and gives a design structure, and defines the components and relationships among them. QSA is expected to enable quantum developers to achieve higher level abstraction.

Quantum Operating System (Quantum OS):

A term introduced as QOS by (Giortamis, Romão, Tornow, & Bhatotia, 2024) in their work titled *"QOS: A Quantum Operating System.",* is defined as the system stack that helps manage quantum resources.

Control:

Control is a layer in the quantum computing stack that involves the translation of quantum operations, for example gates application into analog signals. It acts as a mediator between the quantum and classical worlds.

Universal Quantum Compiler:

We define a universal quantum compiler as a compiler that translates high level quantum programs to control level (quantum operations to analog signals) and then map them to hardware level pulses to manipulate qubits

Quantum Software Development Lifecycle (QSDLC):

As briefly described in the first section of this whitepaper, QSDLC is a cycle similar to the one we are familiar with in classical software engineering, starting from requirements analysis to maintenance. However, each of these phases itself wouldn't be similar to their classical counterparts due to inherent nature of the science that we use for quantum computation. This means that although we can take inspiration from classical SWE, we must adapt the phases of our cycle, which can position us to create just as reliable, scalable, and efficient software as is possible with classical SWE if not better.

Stakeholders:

Current state-of-the-art Quantum Computing: Software, Hardware, Stakeholders

I. Quantum Hardware – Technologies & Vendors

Developers of quantum hardware and computing technologies face several critical challenges, including scalability, error correction, decoherence, temperature requirements, and system integration. Effectively addressing these challenges is vital for advancing quantum computing towards practical and scalable applications. This section will discuss various quantum computing hardware technologies and highlight companies focussing on developing these technologies and associated quantum processing units (QPUs).

Superconducting

Superconducting qubit-based systems utilize superconducting circuits to generate and control qubits, which are the fundamental units of quantum information. Operating at ultra-low temperatures to achieve superconductivity, these circuits enable precise manipulation of quantum states. Superconducting qubits are favoured in quantum computing for their scalability, rapid gate operations, and relatively

well-characterized error rates. Companies in this field, including IBM, Google, Intel, Rigetti, and Quantum Circuits Inc., are focussing on the development of superconducting quantum hardware.

Trapped ion

Trapped ion quantum computers utilize electrically charged ions confined within electromagnetic traps as qubits. These ions are precisely manipulated using lasers to execute quantum operations. This technology is distinguished by its high precision and low error rates, attributed to the isolation of the ions. Trapped ion systems are recognized for their robust quantum gate operations and extended coherence times, positioning them as a prominent candidate for scalable quantum computing. Companies such as IonQ and Quantinuum focus on developing trapped ion-based quantum computing technologies.

Annealing

Quantum annealing is not per se a qubit technology but is a certain way of using the technology which is employed in quantum computers to address optimization problems by identifying the lowest energy state of a system. This approach involves initializing a quantum system in a superposition of states and progressively evolving it to minimize a specified objective function. Unlike gate-based quantum computing, quantum annealing is tailored for optimization tasks and is particularly effective for navigating complex, rugged problem landscapes. D-Wave Systems is one of the companies in this domain, focussing on the development of quantum annealing hardware.

Photonic

Photonic quantum computers use photons to represent and process quantum information. They leverage the quantum properties of light, such as superposition and entanglement, to perform computations. This technology benefits from high-speed data transmission and low error rates, as photons are less susceptible to decoherence compared to other qubit types. Companies like Xanadu and PsiQuantum are focussing on developing photonic quantum computing technologies.

NV centers in diamonds

Quantum computing technologies using nitrogen-vacancy (NV) centers in diamonds utilize these defects in diamond crystals as qubits. NV centers exhibit quantum properties that can be manipulated using optical and microwave techniques. This approach benefits from high precision, long coherence times, and operational stability at room temperature. Companies like Quantum Brilliance, Infleqtion are focussing on this technology for its potential in robust and scalable quantum computing.

Quantum Simulators

Although quantum simulators are not technically "quantum hardware", we would like to give it a special mention.

Quantum simulators are classical computing systems engineered to emulate quantum computers and their algorithms. These simulators enable researchers to test and develop quantum algorithms on classical hardware by replicating quantum phenomena. Although they cannot fully capture the performance of actual quantum computers, they offer critical insights into quantum algorithms and assist with debugging and optimization. Quantum simulators are vital for bridging the gap between theoretical quantum computing and practical hardware implementation. Companies such as NVIDIA and Atos are focussing on quantum simulation technologies.

Other notable quantum hardware use technologies like neutral atoms and NMR (nuclear magnetic resonance) to perform quantum computation.

II. Quantum Software – Proprietary and Open Source

Proprietary quantum software

Proprietary quantum software development toolkits for creating quantum software and algorithms include, Rigetti Forest [6], Hewlett Packard Enterprise (HPE) Quantum Simulator [7], D-Wave Ocean [8], IonQ's Quantum Cloud [9], Classiq SDK (offers both enterprise and open source) [10], QC Ware's Promethium [11], Multiverse Computing's Singularity SDK [12] and others. Some of the below stated open-source quantum software like Pennylane by Xanadu, Qiskit IBM Quantum Experience, Cirq for particularly Google Quantum AI, also offer some advanced features as proprietary components, that help build upon their open-source offering for more complex, often business-scale applications.

Open-Source Quantum Software

In addition, several open-source toolkits support the development of quantum software and algorithms. These include Qiskit (IBM) [13], Cirq (Google) [14], Microsoft Quantum Development Kit (QDK) [15] Q-CTRL Python Open Controls [16], Quantify by Qblox and Orange Quantum Systems [17], Intel Quantum Simulator [18] Berkeley Quantum Synthesis Toolkit [19], SILQ (ETH Zürich) [20], Qibo [21], Bluqat [22], TensorFlow Quantum [23], Qbsolv (D-Wave) [24], LIQUi|> (Microsoft) [25], PennyLane by Xanadu [26], Pasqal's pasqal-cloud [27], Rigetti pyquill [28] and among others.

It's worth noting that some of these quantum software service providers interestingly have both proprietary, enterprise solutions and open-source SDKs, allowing not only researchers and academics but also the general quantum enthusiast to build solutions using their quantum software tools. There exist additional hardware dependencies which are imposed on the software tool to restrict users to execute only on their quantum hardware offerings, which make the lines of open-source versus proprietary blurry and often comes with a cost to execute on their hardware, when building more scalable applications and working on larger-scale projects, which necessitate large number of qubits.

III. Universities and Research Institutions: Quantum Algorithms, Quantum Computing, Quantum Software Engineering

Globally, several universities are at the forefront of quantum algorithms and software development. These institutions are advancing quantum computing technologies through extensive research and development in quantum algorithms and software. Notable examples include:

- University of California, Berkeley (quantum algorithms, quantum software development, quantum error correction) [29]
- University of Cambridge (quantum computing algorithms, quantum information, and related fields) [30]
- Massachusetts Institute of Technology (MIT) (quantum computing, quantum algorithms, quantum machine learning, quantum simulations, and quantum complexity theory) [31]
- Harvard University (quantum algorithms, quantum error correction, and quantum information processing) [32]
- University of Oxford (quantum algorithms, quantum information theory, and quantum software development) [33]

- University of Waterloo (quantum algorithms and software development) [34]
- University of Toronto (quantum algorithms, quantum machine learning, and quantum software) [35]
- University of Copenhagen (quantum computing theory and algorithms) [36]
- ETH Zurich (quantum algorithms, quantum information theory, and quantum software development) [37]
- California Institute of Technology (Caltech) (quantum algorithms, quantum error correction, and quantum information theory) [38]
- Carnegie Mellon University (Software Engineering Institute) (quantum computing, quantum software architecture) [39]

These universities, among others, are significantly contributing to the progress and innovation in the field of quantum computing, quantum algorithms, and quantum software development.

IV. Quantum Computing – Practical Applications and Industry Specific QSWE

Quantum computing offers substantial potential across a variety of industrial applications, including;

Pharmaceuticals and Healthcare: Quantum computing has the potential to significantly accelerate drug discovery and development by simulating molecular interactions with unprecedented speed and accuracy. This capability could lead to substantial advancements in personalized medicine and the development of new therapies. It's a known fact that quantum computing promises substantial advantages for applications and challenging open scientific problems like drug discovery, protein folding, genetics research, and more biomedical and pharma applications. As a prominent example, quantum simulation has been found to have the potential to enable faster and more accurate characterizations of molecular systems than existing quantum chemistry methods. [40]

Finance: Quantum algorithms have the potential to improve risk analysis, optimize trading strategies, and enhance portfolio management by processing complex financial models and large datasets with greater efficiency than classical computers. [41]

Materials Science: Quantum computing facilitates the discovery and design of new materials with specific properties, such as high-temperature superconductors or advanced polymers, by enabling the simulation of atomic and molecular structures. [42]

Logistics and Supply Chain: Quantum algorithms can optimize complex supply chain and logistics challenges, including route planning and inventory management, thereby enhancing efficiency and reducing costs. [43]

Cryptography: Quantum computing has the potential to compromise existing cryptographic protocols, necessitating the development of quantum-resistant encryption methods to safeguard sensitive data. [44]

Artificial Intelligence and Machine Learning: Quantum computing can advance machine learning algorithms by enabling faster data processing and more efficient training of complex models, resulting in enhanced pattern recognition and decision-making capabilities. [45]

Energy Sector: Quantum computing has the potential to optimize energy grids, enhance materials for energy storage, and simulate chemical processes, thereby improving energy production and management. [46]

Chemistry: Quantum simulations can enhance our understanding of chemical reactions and processes, facilitating advancements in fields such as catalysis and synthetic chemistry. [47]

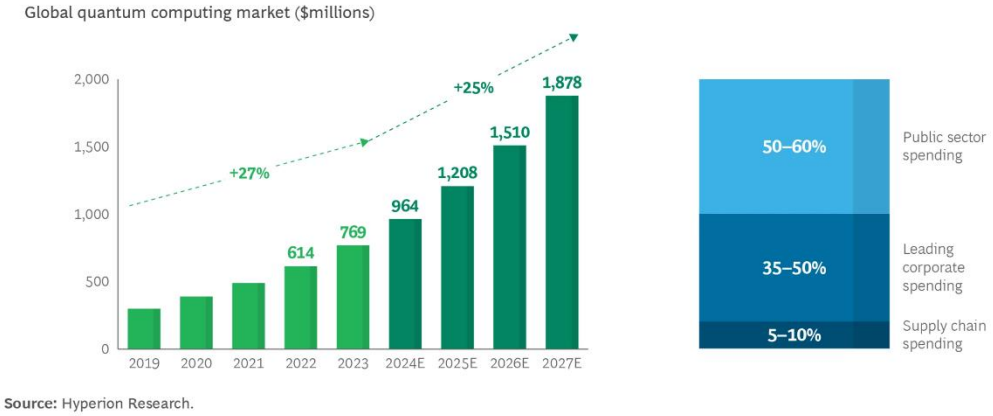
Aerospace and Defence: Quantum computing aids aerospace and defence by optimizing satellite networks, enhancing cryptographic security, developing advanced materials, improving simulations for design, refining signal processing, and solving complex optimization problems more efficiently. [48]

These applications underscore the transformative potential of quantum computing across diverse industries, promising significant enhancements in efficiency, accuracy, and innovation.

V. Governments as Stakeholders in Quantum Computing

Governments around the world are making substantial investments and taking active roles in the development of quantum computing. Notable examples include:

Exhibit 4 - Public Sector Spending Will Continue to Support the Growth of the Quantum Computing Market



Source: Hyperion Research.

Fig 2: Public Investments in Quantum Computing. Retrieved from https://www.bcg.com/publications/2024/long-term-forecast-for-quantum-computing-still-looks-bright

United States: The U.S. government has committed significant funding through initiatives such as the National Quantum Initiative Act and investments by agencies like the Department of Energy (DOE) and National Science Foundation (NSF). [49]

European Union: The EU has launched the Quantum Flagship program, a 10-year initiative aimed at advancing quantum technologies and fostering collaboration across member states. [50]

As a part of EU, The Netherlands government is actively supporting quantum computing through substantial funding for research and development to advance quantum technologies and foster collaboration among academic institutions, industry, and government agencies. [51]

China: China is investing heavily in quantum research through initiatives like the National Laboratory for Quantum Information Sciences and various government-backed research projects. [52]

Canada: The Canadian government supports quantum research through programs such as the Quantum Flagship and investments in institutions like the University of Waterloo's Institute for Quantum Computing. [53]

United Kingdom: The UK government's National Quantum Technologies Programme focuses on funding research and development in quantum technologies, including computing. [54]

Australia: Australia is investing in quantum research and development through initiatives like the Australian Research Council (ARC) and collaborations with institutions like the University of Sydney. [55]

Singapore: Singapore is actively advancing quantum computing through significant investments in research and aims to foster innovation, enhance its technological capabilities, and contribute to global advancements in quantum computing. [56][57]

These investments reflect the strategic importance of quantum computing in advancing technological capabilities and maintaining global competitiveness.

Inspiration from Classical Software Engineering

As it is the case with classical software engineering, envisioning a quantum software stack that might range from programming languages to compilers to operating systems to hardware control would serve as a good starting point.

An interesting aspect of classical software engineering for quantum software engineering is the avenue of formal verification, which is a challenging sub-component as much as an important one. This sub-component is used to check if programs are correct using model checking, formalisms, and techniques

like theorem proving. Verification of quantum programs becomes particularly important in the quantum setting due to the sheer difficultly in programming complex algorithms “correctly”, given resource-constrained and error-prone quantum hardware devices. [58]

One of the primary differentiators between quantum computing and classical computing is the architecture of hardware devices used for computation. Classical computers have evolved over the years from the simple abacus to Charles Babbage’s infamous Analytical Engine to the modern computers that we use today. It will not be wrong to say that the evolution has stabilized over these decades to formalize the present universal modern computer. However, this is not yet the case for quantum computers. Researchers and engineers building such quantum devices come from different schools of thought, paving the way for multiple quantum architectures which mainly vary depending on the type of qubits used.

Moreover, in contrast to quantum software engineering, clear standards have been developed, evolved, and accepted in classical software engineering over the years since the 1960s. It’s noteworthy that the stabilization and establishing of standards has taken years of research and efforts which allows classical software with high levels of quality to be written, shipped, and maintained today. ISO/IEC TR 19759:2015 highlights the best practices in a Software Development Life Cycle starting from Software Requirements Analysis, Software Design, Software Construction, Software Testing and Maintenance, which has enabled seamless development of classical software applications. While a model for a QSWE SDLC might not be as straightforward as the classical SDLC, ISO/IEC TR 19759:2015 [59] inspires similar standardization efforts for quantum software applications.

Years of research and efforts to create, implement, and uphold a broad range of these software development standards, driving the creation, deployment, and maintenance of software applications by standardization organizations like the ISO (International Organization for Standardization), IEEE (Institute of Electrical and Electronics Engineers), and the IEC (International Electrotechnical Commission) on software development standards have culminated into an universal approach to classical software engineering. Such standardizations have enabled programming languages to act in a hardware agnostic manner, allowing them to operate on several different hardware without much changes.

Moreover, this has given a structured approach to creating quality software applications which the industry can follow, to support features like reusability, reliability, scalability, portability, and compatibility, which enhance the calibre of the software. Some important ISO standards have been established for classical software engineering which we are discussing in ISO/IEC/IEEE 15288:2023 [60]. Systems and software engineering is a standard in classical software engineering that has given an universal framework for the creation and sustenance of software applications throughout their life which included the conceptual model of the software’s architecture that addresses the higher level description

of processes involved. As established through ISO IEC 42010 [61], Software Architecture gives a bird's eye view and blue print of the software application that describes the components and their connectors while abstracting out the lower-level implementation aspects. Often, the Software Architecture is expected to include the architecture viewpoints and description languages while also solidifying programming conventions and other common methodologies of describing software architectures.

Since the quantum computing industry has already started to leverage the power of quantum mechanics for more efficient computation of certain special problems, it only becomes crucial now to research the best practices for QSWE. Although the risk maturity is relatively low, with the field of quantum computing itself in its infancy stages, efforts towards proposing approaches to universalize QSWE are a promising bet for shorter and longer terms. Quantum computing is the future and it's going to be possible through Quantum Software Engineering. Thus, it's crucial to focus efforts on standardization and universalization in the present.

The short-term benefits may include cost cutting and reducing development hours for startups and quantum software development companies as seen in classical software engineering; but also the prospects of filling the three major gaps in this field – 1) Lack of a talented workforce, 2) Same piece of code often cannot be executed on multiple hardware architectures and 3) Too many hardware, programming language, and vendor dependent quantum computing SDKs and frameworks to learn.

Quantum computing's current limitation of only being capable of hybrid classical-quantum execution rather than a purely quantum computational model is to be taken into account while proposing such standards. It's becoming increasingly important to introduce such standards for quantum software engineering although it's currently only an emerging area of research within the field of quantum computing. This paper strongly invites and urges the stakeholders - Standardization organizations (NIST, ISO, IEEE, IEC, et al), Academic researchers, Quantum Computing Software Development and Applications companies, to play a role in the path ahead in this unexplored area of research to pave the way for remarkable impacts in the world of quantum computing to solve real-world problems through QSWE.

Like classical computers, quantum computers have hardware and software components. The field of QSWE and Quantum Software Development is an expeditiously evolving field that works towards exploiting the power of quantum computing for various applications. Quantum Software Development also faces many challenges, such as scalability and interoperability, in addition to noise and error correction. These challenges are mainly due to the different types of quantum hardware and technologies prevailing today.

Quantum hardware consists of three vital components. The first and the key one is Quantum Data Plane and is the crucial component that "includes the physical qubits and corresponding structures to hold them together", as defined by Amazon Web Services (AWS). The control and measurement plane turn

the control processor's digital signals, which specify what quantum operations are to be performed, to the analog control signals needed to perform the operations on the qubits in the quantum data plane. The control processor plane operates at a lower level of abstraction and it converts compiled code to commands for the control and measurement layer. Designing and manufacturing the control processor plane is very complex and today it is entirely vendor specific as it involves various technologies. The Quantum Processing Unit (QPU) is developed using, among others, superconducting, trapped ion, photonic, neutral atom and annealing based technologies. Every vendor and technology has its own approach and proprietary software development tool kits which makes quantum software development even more challenging. Though there are benefits to specific applications by choosing a specific technology to design the quantum computing platform, in the long run, standardisation of quantum software development will become even more challenging.

To write a piece of code for a quantum computer, one needs to understand quantum mechanics, mathematics and computer science, and be skilled in programming languages like Python. But the present scenario demands the developer to learn different vendor specific hardware and technologies and also software development toolkits. This paper proposes a standard to be formulated to ensure a quantum software developer need to know only a programming language (Python, for instance), quantum mechanics and linear algebra. A piece of code developed with this knowledge should be able to be executed on all types of quantum computers, irrespective of the QPU and type of technologies used. Present quantum hardware vendors encourage learning on one specific type of hardware and SDK. This implies that it will be very challenging for the quantum software development field and for the industry to have a skilled workforce of quantum computing software developers in general. A quantum software developer should be enabled by standards so that his or her code works on any quantum computer.

This necessitates a standard layer which needs to be adhered by all the quantum computer/QPU manufacturers. This will enable anyone to develop quantum software with only the understanding of quantum mechanics and a standard programming language like Python.

Call for Action

As quantum computing rapidly advances, the development and adoption of quantum software engineering stand as critical pillars in transforming this groundbreaking technology from theoretical promise to practical reality. The time has come to unite our efforts and drive a concerted push towards the universalization of quantum software engineering. We strongly urge stakeholders of the quantum computing ecosystem to come together in this important effort of universalization and development of standards for quantum software engineering. We urge standard organizations, research institutions, quantum software vendors, quantum hardware vendors, and universities to collaborate in establishing comprehensive and cohesive frameworks that will guide the development, evaluation, and integration

of quantum software. Such frameworks are essential to ensuring interoperability, reliability, and security across diverse quantum computing platforms. Call for action from identified stakeholders is summarized in the table below.

Stakeholders	Call for Action
Standardization Organizations	Create and endorse standardization practices for industry consistency and benchmarking.
Research Institutions	Prioritize efforts towards implementation of quantum algorithms using the best practices and contributing to addition of new best practices.
Quantum Software Vendors (QSV)	Adopt and advocate best practices for building innovative solutions.
Quantum Hardware Vendors (QHV)	Collaborate with QSV to embody and complement best practices.
Universities	Integrate Quantum Software Engineering in Higher Education specifically aimed at creating the quantum workforce.

Table 1: Call for Action

Standard Organizations: We call on standardization organizations to lead the effort in creating and endorsing standardized practices and protocols for quantum software development whose role is crucial in setting benchmarks that will foster consistency and facilitate the broader adoption of quantum technologies.

Research Institutions: We encourage research institutions to prioritize research that addresses both the theoretical and practical challenges of quantum software engineering whose innovative work will pave the way for breakthroughs that will shape the future of quantum computing. Active involvement in structured the best practices to help influence the decisions or policies made by standardization body will add immense value and bridges any technical gaps.

Quantum Software Vendors: We urge quantum software vendors to adopt and advocate for best practices in quantum software engineering. By aligning with emerging standards and contributing to developing robust software tools, software vendors can help accelerate the deployment of practical quantum solutions. Mostly, they act as adopters of best practices although at this stage, may also have secondary influence in the standardization bodies policies although some might try to push their technology to be the norm, as it has been the case in universalization of any other technology. On the other hand, their on-the-ground expertise in implementing working quantum software tools or algorithms would certainly help in closing the loop in the context of how effective it actually might be to adopt certain best practices and whether it is actually feasible to apply for implementation of real-world applications.

Quantum Hardware Vendors: The collaboration of quantum hardware vendors with software vendors is vital in ensuring that quantum software and hardware are designed to complement each other seamlessly. This is possible by engaging in dialogue with software developers to address compatibility issues and optimize performance.

Universities: We appeal to universities and institutions of higher learning to integrate quantum software engineering into your curriculum and research agendas. By equipping the next generation of engineers and scientists with the knowledge and skills needed, universities will drive innovation and ensure a continuous pipeline of talent.

Together, this council urges the stakeholders to work with us towards a unified approach that will not only enhance the efficacy of quantum software but also drive its widespread adoption. The future of quantum computing depends on our collective commitment to advancing this critical field. We stand on the brink of a new era in technology—let us seize this opportunity to lead with vision and purpose.

References:

[1] Akbar, M. A., Khan, A. A., Mahmood, S., & Rafi, S. (2022). *Quantum software engineering: A new genre of computing*. arXiv. https://doi.org/10.48550/arXiv.2211.13990

[2] Stepney, S., Braunstein, S. L., Clark, J. A., Tyrrell, A., Adamatzky, A., Smith, R. E., Addis, T. Johnson, C., Timmis, J., Welch, P., Milner, R., & Partridge, D. (2005). Journeys in non-classical computation I: A grand challenge for computing research. *International Journal of Parallel, Emergent and Distributed Systems*, *20*(1), 5–19. https://doi.org/10.1080/17445760500033291

[3] Zhao, J. (2020). Quantum Software Engineering: Landscapes and Horizons. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2007.07047

[4] Piattini, M., Peterssen, G., & Pérez-Castillo, R. (2020). Quantum computing. *Software Engineering Notes*, *45*(3), 12–14. https://doi.org/10.1145/3402127.3402131

[5] Liu, J., & Yang, L. (2023). Quantum software engineering: A review and future directions. *Journal of Computing and Information Science in Engineering, 23*(3), 031103. https://doi.org/10.1007/s10515-023-00389-7

[6] Rigetti Computing. *Forest SDK documentation*. Retrieved from https://www.rigetti.com/forest

[7] Hewlett Packard Enterprise. *Quantum Simulator*. Retrieved from https://www.hpe.com/us/en/solutions/quantum-computing.html

[8] D-Wave Systems Inc. *Ocean software documentation*. Retrieved from https://docs.ocean.dwavesys.com

[9] IonQ. *IonQ Quantum SDK*. Retrieved from https://ionq.com

[10] Classiq. *Classiq SDK documentation*. Retrieved from https://www.classiq.io

[11] QC Ware. *Promethium documentation*. Retrieved from https://www.qcware.com/promethium

[12] Multiverse Computing. *Singularity SDK documentation*. Retrieved from https://www.multiversecomputing.com/singularity-sdk

[13] IBM. *Qiskit*. GitHub. Retrieved from https://github.com/Qiskit/qiskit

[14] Google. *Cirq*. GitHub. Retrieved from https://github.com/quantumlib/Cirq

[15] Microsoft. *Q#*. GitHub. Retrieved from https://github.com/microsoft/qsharp/tree/main

[16] Q-CTRL. *Q-CTRL Python Open Controls.* GitHub. Retrieved from https://github.com/qctrl/python-open-controls

[17] Quantify OS. *Quantify Core GitHub repository*. Retrieved from https://github.com/quantify-os/quantify-core

[18] Intel. *Intel Quantum Simulator GitHub Repository.* Retrieved from https://github.com/intel/intel-qs

[19] University of California, Berkeley. *BQSKit: The Boulder Quantum Software Toolkit GitHub repository*. Retrieved from https://github.com/BQSKit/bqskit

[20] ETH Zurich. *Silq.* GitHub. Retrieved from https://github.com/eth-sri/silq

[21] Qibo. *Qibo: A quantum computing simulation library in Python GitHub Repository.* Retrieved from https://github.com/qiboteam/qibo

[22] Blueqat. *Blueqat bqcloud.* GitHub. Retrieved from https://github.com/Blueqat/bqcloud

[23] Google. *TensorFlow Quantum: A library for hybrid quantum-classical machine learning.* GitHub. Retrieved from https://github.com/tensorflow/quantum

[24] D-Wave Systems. *QBsolv: A tool for solving large binary quadratic problems.* GitHub. Retrieved from https://github.com/dwavesystems/qbsolv

[25] Microsoft. *StationQ, Liquid: A library for quantum computing.* GitHub. Retrieved from https://github.com/StationQ/Liquid

[26] Xanadu. *PennyLane: A library for quantum machine learning, quantum computing, and quantum chemistry*. GitHub. Retrieved from https://github.com/PennyLaneAI/pennylane

[27] Pasqal. *Pasqal cloud*. GitHub. Retrieved from https://github.com/pasqal-io/pasqal-cloud

[28] Rigetti Computing. *pyquil: A Python library for quantum programming using Quil*. GitHub. Retrieved from https://github.com/rigetti/pyquil

[29] Berkeley Quantum Information and Computation Center. *Berkeley Quantum Information and Computation Center*. Retrieved from https://vcresearch.berkeley.edu/research-unit/berkeley-quantum-information-and-computation-center

[30] University of Cambridge. *Centre for Quantum Information and Foundations*. Retrieved from https://www.qi.damtp.cam.ac.uk/centre-quantum-information-and-foundations

[31] MIT Center for Quantum Engineering. CQE *Research*. Retrieved from https://cqe.mit.edu/research/

[32] Harvard University. *Harvard Quantum Initiative*. Retrieved from https://quantum.harvard.edu/

[33] Oxford University. *Oxford Quantum Information Institute.* Retrieved from https://www.oqi.ox.ac.uk/home

[34] University of Waterloo. *Institute for Quantum Computing.* Retrieved from https://uwaterloo.ca/institute-for-quantum-computing/research

[35] University of Toronto. *The Matter Lab*. Retrieved from https://www.matter.toronto.edu/basic-content-page/quantum-computing

[36] University of Copenhagen. *Quantum Research, Faculty of Science.* Retrieved from https://science.ku.dk/english/research/quantum-research-at-science/

[37] ETH Zurich. *Research areas. ETH Zurich Quantum Centre*. Retrieved from https://qc.ethz.ch/the-center/members/research-areas.html

[38] California Institute of Technology. *Quantum information and computation centre*. Retrieved from https://www.cms.caltech.edu/research/quantum-information-and-computation

[39] Carnegie Mellon University. *Software Engineering Institute.* Retrieved from https://www.sei.cmu.edu/our-work/quantum-computing/

[40] Cao, Y., Romero, J., & Aspuru-Guzik, A. (2018). Potential of quantum computing for drug discovery. *IBM Journal of Research and Development, 62*(6), 6:1-6:20. https://doi.org/10.1147/JRD.2018.2888987

[41] Herman, D., Googin, C., Liu, X., & et al. (2023). Quantum computing for finance. *Nature Reviews Physics, 5*, 450–465. https://doi.org/10.1038/s42254-023-00603-1

[42] Haug, T., & Kim, M. S. (2021). Generalized measurements with maximized information gain for quantum state estimation. *Quantum Science and Technology, 6*(5), 055011. https://doi.org/10.1088/2058-9565/ac1ca6

[43] Phillipson, F. (2024). *Quantum computing in logistics and supply chain management: An overview*. arXiv. Retrieved from https://arxiv.org/abs/2402.17520v1

[44] Mavroeidis, V., Vishi, K., Zych, M. D., & Jøsang, A. (2018). The impact of quantum computing on present cryptography. *International Journal of Advanced Computer Science and Applications. ArXiv.* Retrieved from https://arxiv.org/pdf/1804.00200

[45] Gupta, S., Mohanta, S., Chakraborty, M., & Ghosh, S. (2017). Quantum machine learning—Using quantum computation in artificial intelligence and deep neural networks: Quantum computation and machine learning in artificial intelligence. In *Proceedings of the 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON)* (pp. 268-274). IEEE. https://doi.org/10.1109/IEMECON.2017.8079602

[46] Paudel, H. P., Syamlal, M., Crawford, S. E., Lee, Y.-L., Shugayev, R. A., Lu, P., Ohodnicki, P. R., Mollot, D., & Duan, Y. (2022). Quantum Computing and Simulations for Energy Applications: Review and Perspective. *ACS Engineering Au*, *2*(3), 151–196. https://doi.org/10.1021/acsengineeringau.1c00033

[47] McArdle, S., Endo, S., Aspuru-Guzik, A., Benjamin, S., & Yuan, X. (2018). *Quantum computational chemistry*. ArXiV. Retrieved from https://arxiv.org/abs/1808.10402

[48] Landers, V. S. (2024). Quantum technologies for space and aerial vehicles. In H. Jahankhani, S. Kendzierskyj, S. Pournouri, & M. A. Pozza (Eds.), *Space governance: Space law and policy* (pp. 105-128). Springer. https://doi.org/10.1007/978-3-031-62228-1_4

[49] National Science Foundation. *Quantum (Focus areas).* Retrieved from https://new.nsf.gov/focus-areas/quantum

[50] European Commission. *Quantum (Digital strategy).* Retrieved from https://digital-strategy.ec.europa.eu/en/policies/quantum

[51] Delft University of Technology. *The Netherlands and quantum computing*. Retrieved from https://www.tudelft.nl/over-tu-delft/strategie/vision-teams/quantum-computing/impact/the-netherlands-and-quantum

[52] Cheung Kong Graduate School of Business. (2024). *Quantum wars: The global race for quantum supremacy*. Retrieved from https://english.ckgsb.edu.cn/knowledge/article/quantum-wars/

[53] Innovation, Science and Economic Development Canada. *Canada's national quantum strategy*. Retrieved from https://ised-isde.canada.ca/site/national-quantum-strategy/en/canadas-national-quantum-strategy

[54] United Kingdom, National Quantum Computing Centre. *National Quantum Computing Centre*. Retrieved from https://www.nqcc.ac.uk/

[55] Australian Government, Department of Industry, Science, and Resources. (2023). *National quantum strategy*. Retrieved from https://www.industry.gov.au/publications/national-quantum-strategy

[56] Government of Singapore. *National Research Foundation Singapore*. Retrieved from https://www.nrf.gov.sg/

[57] SheQuantum. (2024). *Singapore's quantum computing ecosystem: SheQuantum's exclusive interview with the National Research Foundation, Government of Singapore*. Retrieved from https://shequantum.org/2024/07/23/singapores-quantum-computing-ecosystem-shequantums-exclusive-interview-with-national-research-foundation-government-of-singapore/

[58] Lewis, M., Soudjani, S., & Zuliani, P. (2023). Formal verification of quantum programs: Theory, tools, and challenges. *ACM Transactions on Quantum Computing* https://doi.org/10.1145/3624483

[59] International Organization for Standardization. (2015). *ISO/IEC TR 19759:2015: Software engineering — Guide to the software engineering body of knowledge (SWEBOK)*. Retrieved from https://www.iso.org/standard/67604.html

[60] International Organization for Standardization. (2022). *ISO 50001:2022 Energy management systems—Requirements with guidance for use*. Retrieved from https://www.iso.org/standard/81702.html

[61] International Organization for Standardization. (2009). *ISO 31000:2009 Risk management—Principles and guidelines*. Retrieved from https://www.iso.org/standard/45991.html

[62] Callison, A., & Chancellor, N. (2022). Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond. arXiv preprint arXiv:2207.07094. https://arxiv.org/abs/2207.07094

This whitepaper is a first initiative by the Quantum Software Engineering Standardization Council (QSESC), SheQuantum towards universalization of quantum software engineering.